CLAIMS

What is claimed is:

1 A method for increasing security of a software program by obfuscation of program execution flow, wherein the software program is executed on a computer system that includes a user-level protected mode and a kernel-level unprotected mode, the method comprising the steps of:

(a) identifying critical code segments to be hidden in the software program;

(b) executing non-critical portions of the software program in the user-level protected mode; and

(c) executing the critical code segments within respective exception handlers, thereby hiding execution of the critical code segments from a debugger program.

2 The method of claim 1 wherein step (a) further includes the step of

(i) inserting an exception set-up handler into the software program that sets-up the exception handlers so that the exception handlers will be invoked during program execution.

3 The method of claim 2 wherein step (a) further includes the step:

(ii) inserting an in-line code segment in the software program, wherein the in-line code segment sets up and executes the exception set-up handler during program execution.

4 The method of claim 1 wherein step (a) further includes the step of:

(i) inserting a kernel level driver in the software program that sets-up the exception handlers so that the exception handlers will be invoked during program execution.

5 The method of claim 4 wherein step (a) further includes the step of:

(ii) inserting an in-line code segment in the software program, wherein the in-line code segment invokes the kernel-level driver, thus causing the exception handlers to be invoked during program execution.

6 The method of claim 3 further including the step of: using the set-up handler to initiate a set-up of debug registers, such that the critical code segments in the exception handlers are executed at appropriate times during the program execution.

7 The method of claim 6 further including the step of: executing the in-line code segment prior to a point in the program flow where any of the critical code segments was removed for placement into the exception handlers.

8 The method of claim 7 further including the step of: using the in-line code segment to install the exception set-up handler on an exception handler linked list for a current thread, such that when an exception occurs, the operating system hands control to the exception set-up handler for execution.

9 The method of claim 8 further including the step of: removing the exception set-up handler from the linked list after execution.

10 The method of claim 8 further including the step of: inserting each of the exception handlers into the linked list.

11 The method of claim 10 further including the step of: when an exception is raised to the operating system, handing control down the linked list until one of the exception handlers determines that the exception is one that the exception handler is designed to handle.

12 The method of claim 11 further including the step of: if the exception is one for which the current exception handler was designed to handle, executing the critical code segment included in the current exception handler.

13 The method of claim 12 further including the step of: if the exception is one for which the exception set-up handler was designed to handle, setting a return address for the exception set-up handler when the exception processing completes.

14 The method of claim 13 further including the step of: modifying debug registers during execution of the exception handlers such that any number of exception handlers may be daisy chained.

15 A method for increasing security of a software program by obfuscation of the software program execution flow, the method comprising the steps of:

(a) partitioning the software program into code segments, each code segment having a particular location within the program execution flow;

(b) identifying one or more critical code segments;

(c) encapsulating the critical code segments in respective exception handlers; and

(d) during software program execution, executing the code segments and the exception handlers containing the critical code segments in their respective relative location within the program execution flow.

16 A method for obfuscation of computer program execution flow to increase computer program security, the method comprising the steps of:

(a) breaking the computer code into a plurality of code segments, and identifying critical code segments to be obfuscated;

(b) embedding each of one or more critical code segments within respective first exception handlers;

(c) providing an exception set-up handler to set up operation of the first exception handlers;

(d) embedding an in-line code segment within a first one of the remaining plurality of code segments for setting up and invoking the exception set-up handler.

17 The method of claim 16 wherein step (c) further includes the step of: using the set-up handler to initiate a set-up of debug registers, such that the critical code segments in the exception handlers are executed at appropriate times during the program execution.

18 The method of claim 16 wherein step (d) further includes the step of: executing the in-line code segment prior to a point in the program flow where any of the critical code segments was removed for placement into the exception handlers.

19 The method of claim 18 further including the step of: using the in-line code segment to install the exception set-up handler on an exception handler linked list for a current thread, such that when an exception occurs, the operating system hands control to the exception set-up handler for execution.

20 The method of claim 19 further including the step of: removing the exception set-up handler from the linked list after execution.

21 The method of claim 17 wherein step (b) further includes the step of: inserting each of the exception handlers into the linked list.

22 The method of claim 21 further including the step of: when an exception is raised to the operating system, handing control down the linked list until one of the exception handlers determines that the exception is one that the exception handler is designed to handle.

23 The method of claim 22 further including the step of: if the exception is one for which the current exception handler was designed to handle, executing the critical code segment included in the current exception handler.

24 The method of claim 23 further including the step of: if the exception is one for which the exception set-up handler was designed to handle, setting a return address for the exception set-up handler when the exception processing completes.

25 The method of claim 24 further in cluding the step of: modifying debug registers during execution of the exception handlers such that any number of exception handlers may be daisy chained.

26 The method of claim 16 wherein step (b) further includes the step of providing entry code for the first exception handlers to determine a nature of the exception.

27 The method of claim 26 further including the step of handling the exception if the nature of the exception is applicable to the first exception handler, otherwise handing exception processing to the next available exception handler.

28 The method of claim 16 further including the step of obfuscating the critical code segments to prevent reverse engineering.

29 The method of claim 16 further including the step of obfuscating the critical code segments to hide anti-piracy algorithms.

30 The method of claim 16 further including the step of providing code for the first exception handlers to modify a return address to be used after completion of the exception processing

31 The method of claim 30 further including the step of rearranging non-embedded code segments out of normal execution order, and setting exception addresses and return addresses to ensure proper program sequencing to further obfuscate the program execution flow.

32 The method of claim 16 further including the step of including code within the first exception handlers to modify exception conditions to support future exceptions.

33 The method of claim 32, further including the step of including a plurality of code segments within the first exception handler.

34 The method of claim 33, further including the step of selecting which of a plurality of code segments within the first exception handler should be executed based on the exception information.

35 The method of claim 32, further including the step of embedding one or more first exception handlers into other first exception handlers to further obfuscate the program execution flow.

36 A method for obfuscation of computer program execution flow to increase computer program security, the method comprising the steps of:

(a) partitioning the program into a plurality of non-critical code segments and at least one critical code segment;

(b) obfuscating the critical code segments by embedding each of one of the critical code segments within a respective exception handler;

(c) providing a driver to set up operation of the first exception handlers; and

(d) embedding an in-line code segment within a first non-critical code segment for invoking the driver when the program is executed.

37 The method of claim 36 wherein step (c) further including the step of: using the in-line code segment driver to initiate a set-up of debug registers, such that the critical code segments in the exception handlers are executed at appropriate times during the program execution.

38 The method of claim 37 further including the step of: executing the in-line code segment prior to a point in the program flow where any of the critical code segments was removed for placement into the exception handlers.

39 The method of claim 37 wherein step (b) further includes the step of: inserting each of the exception handlers into a linked list.

40 The method of claim 39 further including the step of: when the exception is raised to the operating system, handing control down the linked list until one of the exception handlers determines that the exception is one that the exception handler is designed to handle.

41 The method of claim 40 further including the step of: if the exception is one for which the current exception handler was designed to handle, executing the critical code segment included in the current exception handler.

42 The method of claim 41 further including the step of: if the exception is one for which the exception set-up handler was designed to handle, setting a return address for the exception set-up handler when the exception processing completes.

43 The method of claim 42 further including the step of: modifying debug registers during execution of the exception handlers such that any number of exception handlers may be daisy chained.

44 The method of claim 36 wherein step (b) further includes the step of providing entry code for the first exception handlers to determine a nature of the exception.

45 The method of claim 44, further comprising the step of handling the exception if the nature of the exception is applicable to the first exception handler, otherwise handing exception processing to the next available exception handler.

46 The method of claim 36, further comprising the step of selecting from the plurality of code segments the segments to obfuscate within the first exception handlers based on the value of obfuscating the algorithms within the segment.

47 The method of claim 42, further comprising the step of providing code for the first exception handlers to modify the return address to be used after completion of the exception processing

48 The method of claim 47, further comprising the step of rearranging the remaining non-embedded code segments out of normal execution order, and setting exception addresses and return addresses to ensure proper program sequencing to further obfuscate the program execution flow.

49 The method of claim 42, further comprising the step of including code within the first exception handlers to modify the exception conditions to support future exceptions.

50 The method of claim 49, further comprising the step of including a plurality of code segments within the first exception handler.

51 The method of claim 50, further comprising the step of selecting which of a plurality of code segments within the first exception handler should be executed based on the exception information

52 The method of claim 36, further comprising the step of including additional, unrelated driver code within the driver to further obfuscate the operation of the program execution flow.

53 The method of claim 42, further comprising the step of setting up exceptions to occur within one or more first exception handlers, to be handled by other first exception handlers, to further obfuscate the program execution flow.

54 The method of claim 53, further comprising the step of repeating the embedded exceptions within exception handlers such that the required exception processing occurs at a plurality of exception processing levels, to further obfuscate the program execution flow.

55 A computer-readable medium containing a software program that is to be executed on a computer system that includes a user-level protected mode and a kernel-level unprotected mode, wherein the software program contains critical code segments to be obfuscated during program execution flow to increase security of the software program, the software program comprising instructions for:

(a) executing non-critical portions of the software program in the user-level protected mode; and

(b) executing the critical code segments within respective exception handlers, thereby hiding execution of the critical code segments from a debugger program.

56 The computer-readable medium of claim 55 wherein instruction (a) further includes the instruction of

(i) invoking an exception set-up handler within the software program that sets-up the exception handlers so that the exception handlers will be invoked during program execution.

57 The computer-readable medium of claim 56 wherein instruction (a) further includes the instruction:

(ii) inserting an in-line code segment in the software program, wherein the in-line code segment sets up and executes the exception set-up handler during program execution.

58 The computer-readable medium of claim 55 wherein instruction (a) further includes the instruction of:

(i) invoking a kernel level driver in the software program that sets-up the exception handlers so that the exception handlers will be invoked during program execution.

59 The computer-readable medium of claim 58 wherein instruction (a) further includes the instruction of:

(ii) invoking an in-line code segment in the software program, wherein the in-line code segment invokes the kernel-level driver, thus causing the exception handlers to be invoked during program execution.

60 The computer-readable medium of claim 57 further including the instruction of: causing the set-up handler to set-up debug registers, such that the critical code segments in the exception handlers are executed at appropriate times during the program execution.

61 The computer-readable medium of claim 60 further including the instruction of: executing the in-line code segment prior to a point in the program flow where any of the critical code segments was removed for placement into the exception handlers.

62 The computer-readable medium of claim 61 further including the instruction of: causing the in-line code segment to install the exception set-up handler on an exception handler linked

list for a current thread, such that when an exception occurs, the operating system hands control to the exception set-up handler for execution.

63 The computer-readable medium of claim 62 further including the instruction of: removing the exception set-up handler from the linked list after execution.

64 The computer-readable medium of claim 62 further including the instruction of: inserting each of the exception handlers into the linked list.

65 The computer-readable medium of claim 64 further including the instruction of: when the exception is raised to the operating system, and control is passed down the linked list to each of the exception handlers, determining that the exception is one that a current exception handler is designed to handle.

66 The computer-readable medium of claim 65 further including the instruction of: if the exception is one for which the current exception handler was designed to handle, executing the critical code segment included in the current exception handler.

67 The computer-readable medium of claim 66 further including the instruction of: if the exception is one for which the exception set-up handler was designed to handle, setting a debug register with a return address for the exception set-up handler when the exception processing completes.

68 The computer-readable medium of claim 67 further including the instruction of: modifying debug registers during execution of the exception handlers such that any number of exception handlers may be daisy chained.

69 A system for obfuscation of program execution flow, comprising:

a computer system including a processor, memory, an operating system that provides kernel-level and user-level execution modes, and debug resources to support the generation and processing of exceptions at specified addresses; and

an obfuscated program comprising,

a plurality of critical code segments encapsulated within respective exception handlers, and

a set-up handler for setting up the debug registers such that when the computer system executes the program, the exception handlers containing the critical code segments are executed in the kernel level mode.